Thermo Fisher Scientific: Transforming Unstructured Product Data

By: Grace Bae, Anna Tan, Peter Wu, Chenxiang Zhang
Project Advisor: Peter Freeman

Introduction and Background

- There are over 1,000,000 products associated with Thermo Fisher, each of which comes with a list of product specifications and attribute that need to be correctly filtered and categorized for use in the website search process.
- The specifications and attributes for the product data have non standardized formats, typographical errors and unusual phrases.
- Goal: To learn patterns from the product data, identify products that are being incorrectly filtered or categorized, and suggest new features/attributes that would improve the search and filtering process.

Data

For our project, we specifically focused on data pertaining to category attributes which consisted of 208,944 rows and 12 columns. We focused on the “Taxonomy Category” and “Attribute Name” columns with our target column being the “Values” column for text mining and transforming.

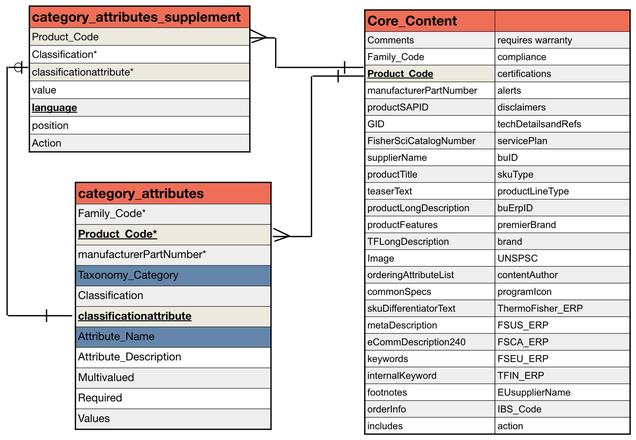
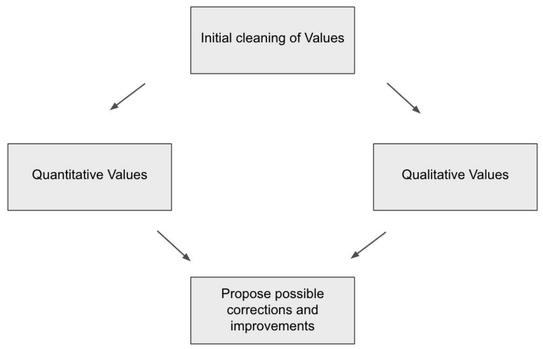


Fig 1: ER-Diagram of data

Our initial EDA consisted of multiple methods such as LDA, word2vec, and text analysis to better understand the patterns and groupings within the “Values” column as a whole and segmented by category and attribute name.

Text Mining & Transformation

Our goal in this step is to develop a framework for dealing with “Values” within both qualitative (“Material”, “Phase”, “Product Line”) and quantitative (“Diameter (Metric)”, “Length (Metric)”, “Max Container Size”) attributes. The proposed pipeline is written to clean/transform existing data, and can help to assist in processing future incoming data in an identical format as well.



Data Preprocessing

First, we began with some initial text cleaning for all values. This included filtering missing values, trimming whitespace, and separating by commas and “and” into separate row values. For example, with a value such as “Accessory, Block”, this had to be separated into two separate values of “Accessory” and “Block”.

Qualitative Values

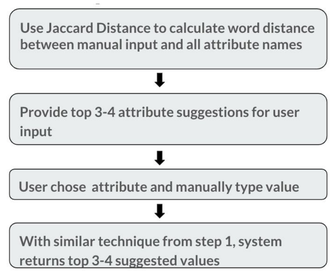
Using the edit distance (Levenshtein distance) with a threshold of distance 1, we were able to identify and fix many of the common typographical errors in the data, such as plural vs. singular words, lowercase and uppercase letters, extra punctuations and other misspellings. After that, we developed string similarity algorithms that combined Levenshtein and Jaro-Winkler distances in order to identify possible groupings among the values, which we propose as a way to find possible filtering categories. In order to be able to propose suggestions beyond the simple Levenshtein distance, we note that we will have to make more generalizations among the strings. Our proposed system thus has the flexibility of managing the trade-off between making very few changes so as to preserve all the information and making more assumptions and generalizations to identify possible groupings, based on a parameter which defines how strict the string similarity must be to output a match.

Quantitative Values

With numerical attributes, because we were dealing with different numerical values and units, we had to use a different approach. We did some additional cleaning by removing any miscellaneous characters such as semicolons and amperstamps. We then separated the numerical values from the units and any additional text and dealt with these two entities separately. We standardized numerical value to be the appropriate integer or float value and to be consistent with a positive sign in front of the value. For units, we used a string similarity algorithm which was based on comparing the set tokens of each string and assigning a new value based on majority rule. This method was used to standardize the units whether this was with casing, spacing, or abbreviations. Lastly, given the new units and additional text, we combined the two parts together to suggest a new final value that would be outputted.

Recommendation System

We have also explored and built a simple recommendation system for the attribute/value mappings.The system first displays all current attributes, and takes an input from user which is supposedly the attribute, then use Jaccard distance to calculate the top 3-4 attribute names with smallest distance.After user choose an attribute, system then again prompts user to type in a desired value, then use the similar technique to identify the top three to four old values--however, the final output will be the corresponding post-filtering values.



Conclusion

Final Statements:

- We explored a variety of text mining and machine learning techniques
- Our pipeline process is able to identify some common incorrectly classified product attribute/value mapping and suggest new values based on the existing data
- Updated about 30% of the values using our method

Suggestions for the Future:

- Implement the pipeline process with more data
- Explore other text related techniques such as neural networks
- Standardize attribute/value formats for future data collections
- Come up with filtering tools connected to the database

Limitations:

- Lack of expert knowledge for some specific product data